UNIVERSITETET
I OSLO

INF1000: Forelesning 6

Klasser og objekter – del 1

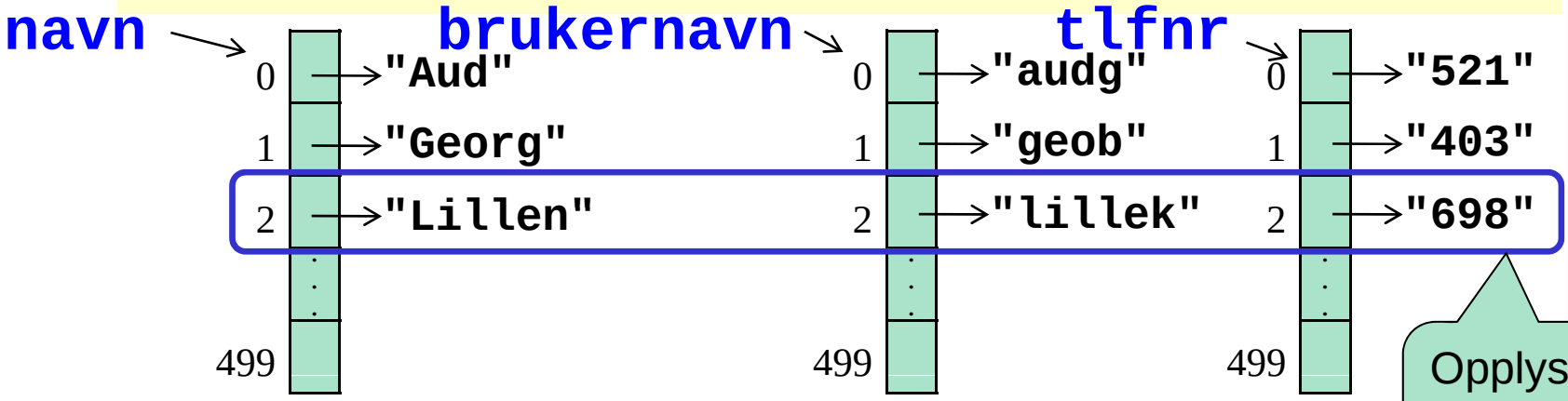
UNIVERSITETET
I OSLO

OBJEKTER SOM EN PROGRAMMERINGS-TEKNIKK

Motivasjon

- Anta at vi ønsker å lage et studentregister hvor vi for hver student lagrer navn, brukernavn og telefonnummer.
- Med arrayer kan dette for eksempel gjøres slik:

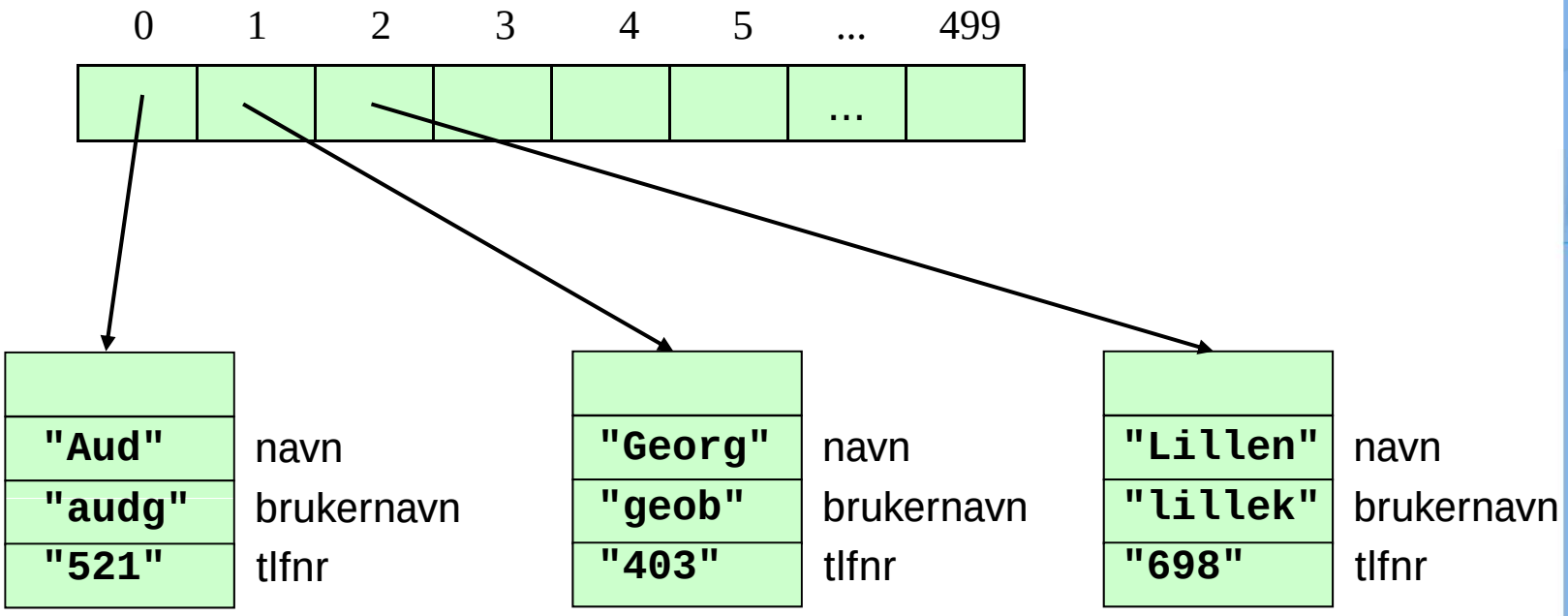
```
int maxAntall = 500;
String[] navn = new String[maxAntall];
String[] brukernavn = new String[maxAntall];
String[] tlfnr = new String[maxAntall];
int antallStud = 0;
```



- Hva er ”galt” med denne måten å gjøre det på?

En bedre løsning: objekter

- Ofte mer naturlig å samle all informasjon om hver student – dette kan gjøres i et ***objekt***.



UNIVERSITETET I OSLO

Aktuelle spørsmål

- Hvordan bestemmer vi hvilke data et student-objekt er i stand til å ta vare på?
- Hvordan oppretter vi et nytt objekt når en ny student skal legges inn i registeret?
- Hvordan legger vi data om studenten inn i objektet?
- Hvordan henter vi data ut av objektet når vi ønsker opplysninger om studenten?
- Hvordan sletter vi et objekt når studenten skal slettes fra registeret?

Klasser – en mal for objekter

- Vi bruker en ***klassedeklarasjon*** til å angi hvilke data et Student-objekt skal kunne ta vare på:

```
class Student {
    String navn;
    String brukernavn;
    String tlfnr;
}
```

- Klassen Student blir da en mal/oppskrift/modell for hva Student-objekter skal inneholde.
- Merk at vi ikke har laget noen objekter enda!

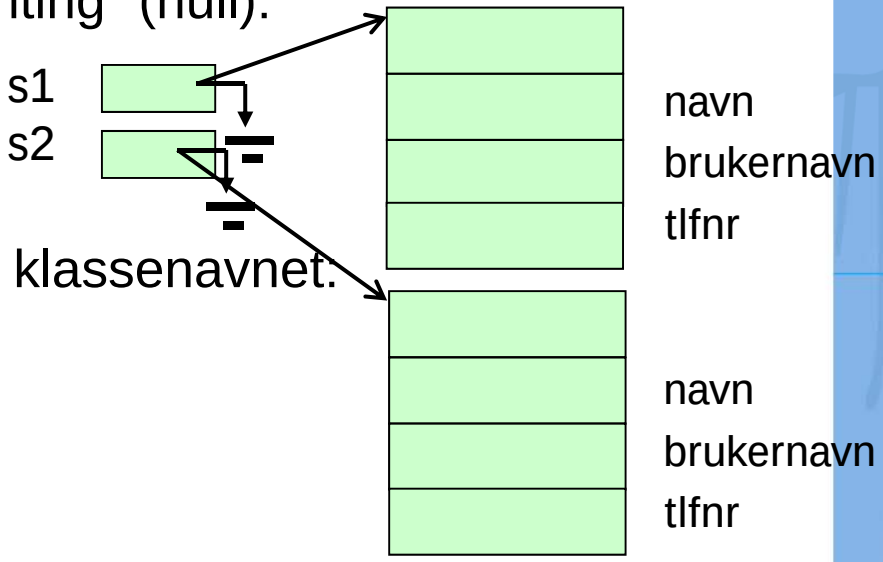
UNIVERSITETET I OSLO

Hvordan lage objekter?

- For å ta vare på/holde orden på objekter trenger vi pekere til dem. Disse deklareres omtrent som andre variable:

```
Student s1, s2;
```

- Foreløpig peker disse på ”ingenting” (null):



- Objekter lages ved å si new på klassenavnet:

```
s1 = new Student();
s2 = new Student();
```

- Vi kan deklarere pekere og opprette objektene samtidig:

```
Student s1 = new Student();
Student s2 = new Student();
```

Hvordan få tilgang til innholdet i objektet?

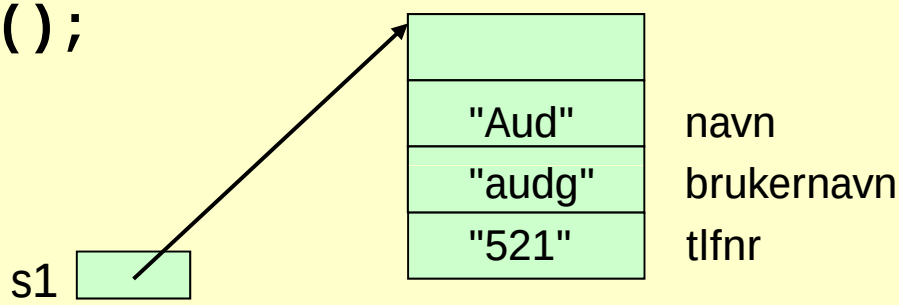
UNIVERSITETET I OSLO

- Variablene i et objekt kan vi få tak i ved å skrive
  `pekernavn.variabelnavn`
- Punktumet leses som ”sin” eller ”sitt”
- Dette kan brukes både til å sette og hente ut verdiene i objektet:

```
Student s1 = new Student();

s1.navn = "Aud";
s1.brukernavn = "audg";
s1.tlfnr = "521";

System.out.println("Navn: " + s1.navn);
System.out.println("Brukernavn: " + s1.brukernavn);
System.out.println("Telefon: " + s1.tlfnr);
```



Metoder i objektene

- Metoder som benytter seg av variablene i et objekt, hører ofte logisk hjemme i dette objektet og kan (bør) deklareres som en del av klassen:

```
class Student {
    String navn;
    String brukernavn;
    String tlfnr;

    void skrivData() {
        System.out.println("Navn: " + navn);
        System.out.println("Brukernavn: " + brukernavn);
        System.out.println("Telefon: " + tlfnr);
    }
}
```

Komplett program

UNIVERSITETET I OSLO

```
class Student {
    String navn, brukernavn, tlfnr;

    void skrivData() {
        System.out.println("Navn: " + navn);
        System.out.println("Brukernavn: " + brukernavn);
        System.out.println("Telefon: " + tlfnr);
    }
}

class StudentRegister {
    public static void main(String[] args) {
        Student s1, s2;
        s1 = new Student();
        s1.navn = "Aud";
        s1.brukernavn = "audg";
        s1.tlfnr = "521";
        s2 = new Student();
        s2.navn = "Georg";
        s2.brukernavn = "geob";
        s2.tlfnr = "403";
        s1.skrivData();
        s2.skrivData();
    }
}
```

```
>java StudentRegister
Navn: Aud
Brukernavn: audg
Telefon: 521
Navn: Georg
Brukernavn: geob
Telefon: 403
```

Studentregister med mange studenter

- Med mange studenter i registeret, kan vi bruke en array av pekere til å ta vare på alle Student-objektene:

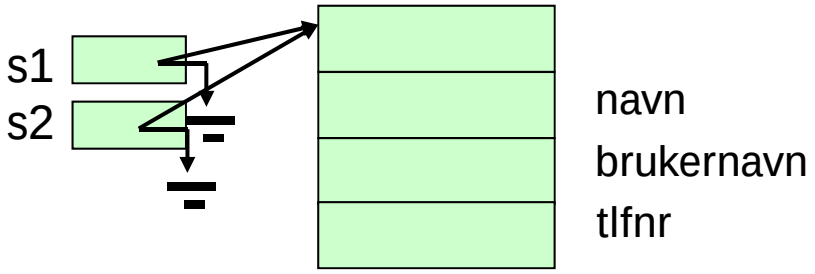
```
class StudentRegister {
    public static void main(String[] args) {
        // Deklarerer datastrukturen
        int maxAntall = 500;
        Student[] studentene = new Student[maxAntall];
        int antallStud = 0;

        // Legger inn testdata om en student
        studentene[antallStud] = new Student();
        studentene[antallStud].navn = "Aud";
        studentene[antallStud].brukernavn = "audg";
        studentene[antallStud].tlfnr = "521";
        // NB: Husk å oppdatere antallStud!
        antallStud++;
    }
}
```

Mer om pekere

- Flere pekere kan godt peke på samme objekt:

```
Student s1, s2;
s1 = new Student();
s2 = s1;
```



- Vi kan sjekke om to pekere peker på det samme eller ikke:

```
if (s1 == s2) {
    // s1 og s2 peker på det samme
}
if (s1 != s2) {
    // s1 og s2 peker ikke på det samme
}
```

Mer om pekere (forts)

- Av og til har vi behov for å teste om en peker virkelig peker på et objekt eller ikke. Vi sammenligner da pekeren med den spesielle null-verdien:

```
if (s1 != null && s1.navn.equals("Aud") {
    // s1 peker på et Student-objekt
    // hvor variabelen navn er lik "Aud"
}
```

- For å fjerne et objekt fra en peker:

```
s1 = null;
```

- Et objekt som ingen pekere peker på, blir tatt av søppeltømmeren – et program som automatisk startes hvis det er lite plass i hukommelsen

UNIVERSITETET
I OSLO

OBJEKTORIENTERING SOM EN TANKEMÅTE

Objekter

- Vi deler often verden inn i enheter når vi snakker om den: Blomster, mennesker, biler, bankkontoer, …
- Slike enheter kaller vi objekter
- Mange av objektene i verden er ganske like (av samme type), men kan skille seg fra hverandre med feks. ulike navn
  - Studentene Kia og Espen
  - To bøker med ulik tittel/forfatter
- Objekter av samme type sier vi tilhører samme klasse
  - Beskrives av de **samme** variablene, men med **ulike** verdier på noen disse
  - Eks: to biler av samme bilmerke men med ulike reg.nummer
- To objekter kan også være helt vesensforskjellige
  - Eks: et tre og en lastebil
  - De er da objekter av **forskjellige** klasser

UNIVERSITETET I OSLO

Hvor mange klasser (og objekter) er det ?

- Hva vi velger å betrakte som et objekt, og hvilke klasser vi bruker for å beskrive en problemstilling, er ikke bestemt på forhånd.
- Hvilke klasser vi bruker til å beskrive et problem, varierer ofte etter hvor mye detaljer vi trenger og hvilke spørsmål vi ønsker å kunne gi svar på:
  - Beskrive problemet med veitrafikk og køer på veiene:
    - Telle antall biler og kanskje skille mellom lastebiler, busser og personbiler
    - Men neppe mer…
  - Beskrive problemet til en bilfabrikk
    - Trenger en detaljert og komplisert beskrivelse av hver bil (bestående av motor, hjul, karosseri, lys,..., hver beskrevet med sin klasse)
    - Mange ulike typer av biler
    - Har da en rekke klasser som hver beskriver sine objekter.

Objektorientert Programmering - I

- Når vi betrakter et problem vi skal lage et datasystem for, gjør vi to avgrensninger:
  1. Vi ser bare på en del av verden (vårt problemområde)
  2. Innenfor problemområdet betrakter og beskriver vi bare det som er der med en viss detaljeringsgrad – bare så mange detaljer vi trenger for å svare på de spørsmål datasystemet skal kunne gi svar på.

- Eks: Hvordan beskrive en student ? Skal vi lage:
  a) Studentregister: bare registrere navn, personnummer, adresse, tidligere utdannelse og kurs (avlagte og kurs vedkommende tar nå)
  b) Legesystem for studenter: ta med svært mange opplysninger om hver student (medisiner, sykdommer, resultat fra blodprøver, vekt...) som vi ikke ville drømme om å ha i et vanlig studentregister

Objektorientert Programmering - II

- Når vi skal lage et programsystem, så skal det i størst mulig grad være en modell av vårt problemområde – en en-til-en kopi:

  Ett objekt i problemområdet (vår avgrensning av verden) skal medføre at det finnes ett objekt i programmet som representerer dette objektet.

  (I tillegg kommer mange klasser og objekter i programmet for å lese fra tastatur og fil, skrive til skjerm,..)

- Eks.: Hver virkelig student skal ha sitt Student-objekt i et studentregister-system.

Eksempel:
Et meget enkelt banksystem

- Vi skal lage et banksystem for å holde orden på alle kontoene til en bank.
- Systemet skal kunne
  - opprette ny konto
  - sette inn penger på en konto
  - ta ut penger fra en konto
  - vise summen av innskudd i banken
- Vi må da bestemme
  - hvilken datastruktur vi skal ha i programmet, dvs
    - hvilke klasser/objekter trenger vi?
    - hvilke opplysninger trenger vi i hver klasse/objekt?
  - hvilke metoder vi trenger, og for hver av metodene hvilken klasse den skal ligge i

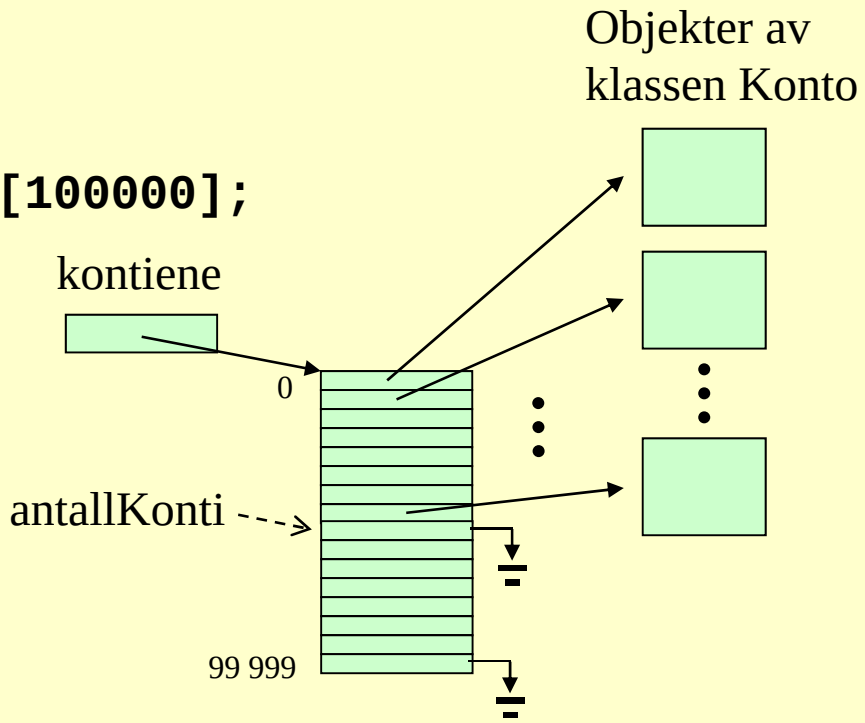
Data i Konto og Bank (en bank har mange konti)

UNIVERSITETET I OSLO

```
class BankSystem {
    public static void main(String[] args) {
        Bank b = new Bank();
        b.navn = "Min Bank";
        b.brukerDialog();
    }
}

class Bank {
    Konto[] kontiene = new Konto[100000];
    int antallKonti = 0;
    String navn;
}

class Konto {
    String navn, adr;
    int kontoNr;
    double saldo = 0.0;
}
```



NB: Skal også ha metoder i Bank og Konto!

Metoden nyKonto

UNIVERSITETET I OSLO

- Skal få data om ny konto fra brukeren, opprette ny konto og legge den inn i systemet.

```
// I klassen Bank:

void nyKonto() {
    // Leser inn data om ny konto
    System.out.print("Navn på ny kontoinnehaver: ");
    String navn = tastatur.inLine();
    System.out.print("Adresse: ");
    String adr = tastatur.inLine();

    // Opretter ny konto
    Konto k = new Konto();
    k.adr = adr;
    k.navn = navn;
    k.kontoNr = antallKonti;

    // Legger ny konto inn i datastrukturen
    kontiene[antallKonti] = k;
    antallKonti++;
}
```

Metoden innskudd

UNIVERSITETET I OSLO

- Skal be om navn på kontoinnehaver og beløp som skal settes inn, finne riktig konto og øke saldoen.

```
void innskudd() {
    Konto k = finnKonto();
       if (k != null) {
       System.out.print("Innskuddsbeløp: ");
       double innskudd = tastatur.inDouble();
       k.settInn(innskudd);
    }
}

Konto finnKonto() {
    System.out.print("Navn på kontoinnehaver: ");
    String s = tastatur.inLine();
    for (int i = 0; i < antallKonti; i++) {
        if (kontiene[i].navn.equals(s)) {
            return kontiene[i];
        }
    }
    System.out.println("Fant ikke kontoinnehaver");
    return null;
}
```

```
// I klassen Konto:
void settInn(double inn) {
    saldo += inn;
}
```

Metoden uttak

UNIVERSITETET I OSLO

- Skal be om navn på kontoinnehaver og beløp som skal tas ut, finne riktig konto, sjekke at det er nok penger på konto og minske saldoen.

```
// I klassen Konto:
boolean taUt(double ut) {
    if (ut > saldo) {
        return false;
    } else {
        saldo -= ut;
        return true;
    }
}
```

```
  void uttak() {
      Konto k = finnKonto();
      if (k != null) {
          System.out.print("Uttaksbeløp: ");
          double uttak = tastatur.inDouble();
          boolean ok = k.taUt(uttak);
          if (!ok) {
              System.out.println("Ikke nok penger på konto.");
          }
      }
}
```

Metoden sumInnskudd

- Skal summere saldoen på alle konti i banken.

```
void sumInnskudd() {
    double sumInnskudd = 0.0;
    for (int i = 0; i < antallKonti; i++) {
        sumInnskudd += kontiene[i].saldo;
    }
    System.out.println("Sum innskudd: " + sumInnskudd);
}
```

UNIVERSITETET
I OSLO

OPPSUMMERING

Oppsummering

- Et objekt er en samling av variabler, pekere og metoder som sammen utgjør en enhet.
- En klasse er en mal/form/oppskrift som vi kan lage objekter med.
- Lager vi to, tre,.. objekter av klassen, får vi to, tre,.. slike kopier
- Vi deler opp programmet vårt i flere klasser
  - Fordi hver programdel (klasse) skal være mulig å holde oversikt over – ikke for stor.
  - Fordi en klasse skal være god modell av en del av det problemområdet vi lager program for.